**XHR对象**

如果只支持IE7以及更高版本只需要var xhr = new XMLHttpRequest();

如要兼容更早版本或者非主流浏览器var xhr = new XMLHttpRequest() || new ActiveXObject();

1. Xhr用法

首先调用方法open（），传3个参数：请求类型‘get’或‘post’，url，和是否异步（传true则异步）。此方法启动一个请求以备发送，如果是同步的请求，js代码会等待ajax请求响应之后再继续执行。

然后send（）方法传入一个参数：发送的数据，如不发送则传null，因为这个参数对有些浏览器来说是必须的。在收到响应之后，响应的数据会自动填充XHR对象的属性。

Xhr对象有4个属性，responseText，responseXML如果内容类型是‘text/xml’或者’application/xml’，这个属性中保存响应数据的XML DOM文档，status响应的http状态码，statusText http状态码的说明。

接收到响应后第一步检查status属性，一般200算成功，responseText内容 已经就绪，此外304也成功。一般检测到200-300和304状态码算成功。有的浏览器会错误得将204状态码报错。

异步请求的时候还需要判断readyState属性，0：创建对象，1：启动，调用open，2：发送，调用send，3：接收部分，4：接收完成。每次改变值的时候都会触发readyStatechange事件。我们只需要在触发事件时检测readyState值是否等于4以及status的值就能知道请求是否完成。

此外还可以接收到响应之前调用abort（）方法来取消异步请求。如xhr.abort()。调用之后会停止触发事件以及不再允许访问任何属性。终止请求之后还应该对xhr对象进行解引用操作。

1. http头部信息

默认情况下，在发送xhr请求还会发送下列头部信息。

使用setRequestHeather（）方法设置请求头，接受2个参数：头部字段的名称和头部字段的值。且要在open之后，send之前设置

getResponseHeader方法能得到单个响应头部信息，getALLResponseHeaders方法能得到一个包含所有头部信息的长字符串。

1. 默认情况下浏览器对post请求 和web表单的请求不会一视同仁，不过可以使用xhr来模仿表单提交。设置头部‘content-Type’为‘application/x-www-form-urlencoded’,send的数据内容也需要进行序列化，见13章serialize函数。

**XMLHttpRequest 2 级**

1. FormData

现代 Web 应用中频繁使用的一项功能就是表单数据的序列化，XMLHttpRequest 2 级为此定义了FormData 类型。[Dome1](dome1.html)

1. 超时设定

12年只有IE8及以上支持超时设定。

xhr.open("get", "timeout.php", true);

xhr.timeout = 1000; //将超时设置为 1 秒钟（仅适用于 IE8+）

xhr.ontimeout = function(){

alert("Request did not return in a second.");

};

xhr.send(null);

timeout 属性设置为 1000 毫秒，意味着如果请求在 1 秒钟内还没有返回，就会自动终止。**请求终止时**，会调用 ontimeout 事件处理程序。若 readyState同时改变为 4 ，会调用 onreadystatechange 事件处理程序。可是，如果在超时终止请求之后再访问 status 属性，就会导致错误。为避免浏览器报告错误，可以将检查 status 属性的语句封装在一个 try-catch 语句当中。

1. overrideMimeType()方法

用于重写 XHR 响应的 MIME 类型。

当服务器返回的 MIME 类型是 text/plain，但数据中实际包含的是 XML。根据 MIME 类型，即使数据是 XML，responseXML 属性中仍然是 null。通过调用 overrideMimeType()方法，可以保证把响应当作 XML 而非纯文本来处理。这个例子强迫 XHR 对象将响应当作 XML 而非纯文。

var xhr = createXHR();

xhr.open("get", "text.php", true);

xhr.overrideMimeType("text/xml");

xhr.send(null);

**进度事件**

loadstart：在接收到响应数据的第一个字节时触发。

progress：在接收响应期间持续不断地触发。

load：在接收到完整的响应数据时触发。

error：在请求发生错误时触发。

abort：在因为调用 abort()方法而终止连接时触发。

loadend：在通信完成或者触发 error、abort 或 load 事件后触发。

每个请求都从触发 loadstart 事件开始，接下来是一或多个 progress 事件，然后触发 error、abort 或 load 事件中的一个

1. load事件

load 事件，用以替代 readystatechange 事件。响应接收完毕后将触发 load 事件，因此也就没有必要去检查 readyState 属性了。会接收到一个 event 对象， target 属性指向 XHR 对象实例，因而可以访问到 XHR 对象的所有方法和属性。然而，并非所有浏览器都为这个事件实现了适当的事件对象。只要浏览器接收到服务器的响应，不管其状态如何，都会触发 load 事件。而这意味着你必须要检查 status 属性，才能确定数据是否真的已经可用了。Firefox、Opera、Chrome 和 Safari 都支持 load事件。开发人员还是要像下面这样被迫使用 XHR 对象变量因为event不被全部浏览器实现。

var xhr = createXHR();

xhr.onload = function(){

if ((xhr.status >= 200 && xhr.status < 300) || xhr.status == 304){

alert(xhr.responseText);

} else {

alert("Request was unsuccessful: " + xhr.status);

}

};

xhr.open("get", "altevents.php", true);

xhr.send(null);